

**(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION
EN MATIÈRE DE BREVETS (PCT)**

**(19) Organisation Mondiale de la Propriété
Intellectuelle**
Burcau international



(43) Date de la publication internationale
3 janvier 2002 (03.01.2002)

PCT

(10) Numéro de publication internationale
WO 02/001346 A3

(51) Classification internationale des brevets⁷ : G06F 9/50, 9/46

(21) Numéro de la demande internationale : PCT/FR01/02019

(22) Date de dépôt international : 26 juin 2001 (26.06.2001)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
1015579 30 juin 2000 (30.06.2000) NL

(71) Déposant (pour tous les États désignés sauf US) : THALES NEDERLAND B.V. [NL/NL]; Zuidelijke Havenweg 40, NL-7554 RR Hengelo (NL).

(72) Inventeur; et
(75) Inventeur/Déposant (pour US seulement) : BELTMAN, E., W. [FR/FR]; Thales Intellectual Property, 13, av. du Prés. Salvador Allende, F-94117 Arcueil Cédx (FR).

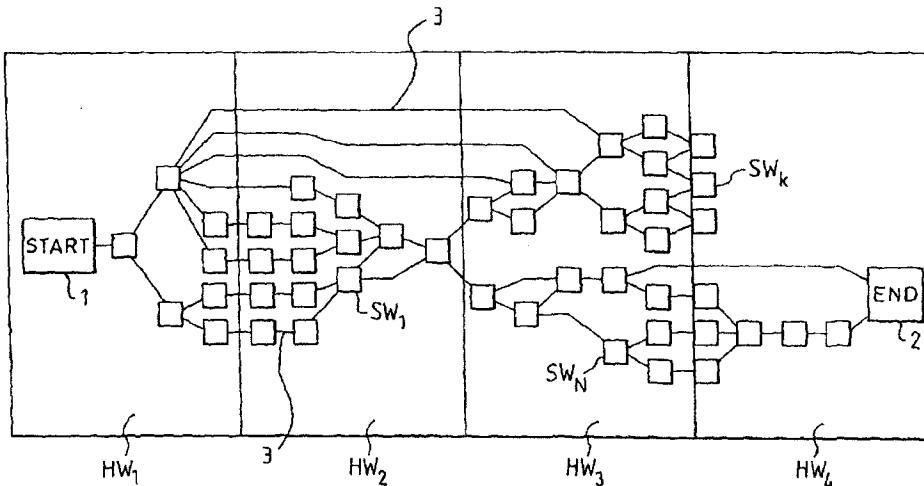
(74) Mandataire : LUCAS, Laurent; Thales Intellectual Property, 13, av. du Prés. Salvador Allende, F-94117 Arcueil Cédx (FR).

(81) États désignés (national) : AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

[Suite sur la page suivante]

(54) Title: METHOD FOR AUTOMATICALLY IMPLANTING SOFTWARE FUNCTIONS ON A SET OF PROCESSORS

(54) Titre : PROCEDE D'IMPLANTATION AUTOMATIQUE DE FONCTIONS LOGICIELLES SUR UN ENSEMBLE DE PROCESSEURS



WO 02/001346 A3

(57) Abstract: The invention concerns a method for automatically implanting software functions on a set of processors. The method comprises at least: a step which consists in breaking down the software functions into elementary tasks (SW₁, SW_k, , SW_N); a step which consists in implanting the elementary tasks on the processors (HW₁, HW₂, HW₃, HW₄); a step which consists in controlling implantation evaluating parameters whereof a list is pre-established, a penalty being assigned when a parameter does not fulfil a given criterion; a step which consists in calculating implantation cost, said cost being the sum of penalties assigned, the selected implantation being based on said cost. The invention is particularly applicable for systems, such as for example radar systems, comprising a large amount of software.

[Suite sur la page suivante]

(19) 日本国特許庁(JP)

(12) 公表特許公報(A)

(11) 特許出願公表番号

特表2004-509386
(P2004-509386A)

(43) 公表日 平成16年3月25日(2004.3.25)

(51) Int.Cl.⁷
G06F 15/177
G06F 9/46F 1
G06F 15/177 674A
G06F 9/46 360Bテーマコード(参考)
5B045
5B098

審査請求 未請求 予備審査請求 有 (全 40 頁)

(21) 出願番号 特願2002-506415 (P2002-506415)
 (86) (22) 出願日 平成13年6月26日 (2001.6.26)
 (85) 翻訳文提出日 平成15年1月6日 (2003.1.6)
 (86) 國際出願番号 PCT/FR2001/002019
 (87) 國際公開番号 WO2002/001346
 (87) 國際公開日 平成14年1月3日 (2002.1.3)
 (31) 優先権主張番号 1015579
 (32) 優先日 平成12年6月30日 (2000.6.30)
 (33) 優先権主張国 オランダ(NL)
 (81) 指定国 EP(AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW

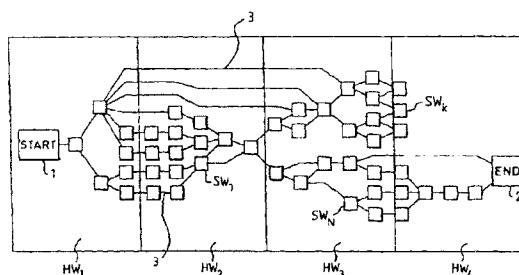
(71) 出願人 501486109
 タレス ネデルラント ペー. フエ.
 オランダ国 エンエル-7550 ヘーデ
 ー ヘンヘロ ザウデライケ ハフェンウ
 エフ 40 ピー. オー. ボックス 42
 (74) 代理人 100109726
 弁理士 園田 吉隆
 (74) 代理人 100101199
 弁理士 小林 義教
 (72) 発明者 ベルトマン, イー., ダブリュー.
 フランス国 エフ-94117 アークイ
 ュ セデックス, アヴェニュー デュ
 プレジダン サルヴァドール アイヤンド
 13, タレス インテレクチュアル
 プロパティ

最終頁に続く

(54) 【発明の名称】複数のプロセッサにソフトウェア機能を自動的に割り当てる方法

(57) 【要約】

本発明は、ソフトウェア機能を複数のプロセッサに自動的に割り当てる方法に関するものである。当該方法は、少なくとも、ソフトウェアの機能を基本タスク ($SW_1, \dots, SW_k, \dots, SW_N$) に分解する過程と、基本タスクをプロセッサ (HW_1, HW_2, HW_3, HW_4) に割り当てる過程と、予め作成したリストに基づいて、所定の条件が満足されないときはペナルティを加算する導入評価パラメータを制御する過程と、導入に関する選択を行うために、パラメータに対応するペナルティの合計で表わされる導入コストを計算する過程とを有する。本発明は特に、例えば、膨大な量のソフトウェアを備えたレーダーシステムのようなシステムに適用することができる。



【特許請求の範囲】

【請求項 1】

ソフトウェアの機能を複数のプロセッサに割り当てる方法であって、少なくとも、
 一ソフトウェアの機能を基本タスク ($S W_1, \dots, S W_k, \dots, S W_N$) に分解し、
 異なる基本タスク相互間のリンクを定義するファイル (32) とプロセッサ ($H W_1, H W_2, H W_3, H W_4$) 相互の接続を定義するファイル (34) を作成する過程と、
 一前記のファイル (32, 34) にしたがって、基本タスクをプロセッサ ($H W_1, H W_2, H W_3, H W_4$) に割り当てる過程と、
 一予め作成したリストを用いて割り当ての評価パラメータをチェックし、所定の評価条件
 が満足されないときは当該割り当てに対してペナルティを加算する過程と、
 一割り当てに対して、対応するペナルティの合計で表わされるコストを計算して、割り当
 ての選別に用いる過程とを有する方法。

10

【請求項 2】

選択された割り当ては、実質的にコストが最小になる割り当てに相当することを特徴とする請求項 1 に記載の方法。

【請求項 3】

前記タスク割り当て過程、評価パラメータのチェック過程およびコスト計算過程は繰り返し実行し、繰り返しによるコストの変化が所定の閾値よりも小さくなつたときにその設定を選択することを特徴とする請求項 1 又は 2 に記載の方法。

20

【請求項 4】

前記繰り返しの第 1 回目の実行の際に、各タスクをプロセッサに対してランダムに割り当て、以降の繰り返しを行う際に、ランダムに選択した 1 つのタスクを別のプロセッサに割り当てる特徴とする請求項 3 に記載の方法。

【請求項 5】

前記評価パラメータはデータフロー、プロセッサの負荷およびタスクの処理時間に関するものであることを特徴とする請求項 1 ないし 4 の何れかに記載の方法。

【請求項 6】

基本タスク ($S W_1, \dots, S W_k, \dots, S W_N$) のそれぞれにソフトウェアコンポーネントが対応し、選択した割り当てにしたがって、コード発生器 (36) がプロセッサのオペレーティングシステム (R T O S) と相互に通信する中間レイヤ (ミドルウェア) (37) を作成し、コード発生器はさらに各ソフトウェアコンポーネントの周囲にソフトウェアレイヤ (38) を作成して、中間レイヤ (37) と相互に通信させて対応するプロセッサによる実効を可能にすることを特徴とする請求項 1 ないし 5 の何れかに記載の方法。

30

【発明の詳細な説明】

【0001】

【技術分野】

本発明は、ソフトウェアの機能を複数のプロセッサに自動的に割り当てる方法に関するものである。本発明は、特に、例えば膨大な量のソフトウェア機能を備えたレーダーシステムのようなシステムに適用することができる。

【0002】

40

【背景技術】

特にレーダーシステムで使用されるソフトウェア機能の量は急速に増大しつつある。同様に、当該システムで使用されるプロセッサの量も増大しつつある。ここで用いられるプロセッサは、例えば信号処理用プロセッサのような形式のものである。アプリケーションによっては数十のプロセッサを必要とすることもある。使用可能なプロセッサに対してソフトウェアを割り当てる仕事は益々時間がかかり、かつ困難になりつつある。さらに、頻繁に行われているようにソフトウェア機能を追加する場合、ソフトウェアまたはハードウェアの基本構造に本質的な変更をせずに複数のプロセッサに対してソフトウェアを再配分することは不可能である。

【0003】

50

したがって、このようなシステムでは、複数のプロセッサに対するソフトウェア機能の割り当ては非常に重要な問題である。ソフトウェア機能の導入に要する時間とその複雑さの問題に加えて、柔軟性の問題もある。実際には、新たなソフトウェア機能を容易に取り込める必要がある。これらの問題は、システムの製造コストの増大につながり、さらに、システムのメンテナンス、テストおよび信頼性に影響を与えることになる。

【0004】

【発明の開示】

本発明の目的は、特に、複数のプロセッサに対するソフトウェアの自動的かつ最適な、つまり単純かつ経済的な割り当てを可能にすることである。この目的のために、本発明は複数のプロセッサに対するソフトウェアの割り当て方法であって、少なくとも、

10

—ソフトウェアの機能を基本タスクに分割して、タスク間のリンクとプロセッサの接続を定義するファイルを作成する過程と、

—前記のファイルにしたがってプロセッサに基本タスクを割り当てる過程と、

—予め作成したリストを用いて割り当ての評価パラメータをチェックし、所定の評価条件が満足されないときは当該割り当てに対してペナルティを加算する過程と、

—割り当てに対して、対応するペナルティの合計で表わされるコストを計算して、割り当ての選別に用いる過程と、

を有する割り当て方法に関するものである。

【0005】

選択された割り当てはコストが最小になるものであることが望ましい。タスクの割り当て過程と、割り当てパラメータのチェック過程と、コストの計算過程とを繰り返し、コストの変動が、例えば2%から3%程度である所定の閾値を下回ったときにソフトウェア機能の設定を行う。評価パラメータは特に、タスクに対するデータフロー、プロセッサの負荷および処理時間に関するものである。

20

【0006】

本発明の最も重要な利点は、システムの種々のタスク割り当てに大きな柔軟性をもたらし、システムの信頼性を向上させ、システムのメンテナンスを容易にし、サブアッセンブリの取り扱いを容易にすることである。

【0007】

本発明の上記以外の特徴と利点は、添付の図面を参照しながら以下の記載によって明らかにする。

30

【0008】

【発明を実施するための最良の形態】

図1に、N個のソフトウェア機能 $SW'_1, \dots, SW'_k, \dots, SW'_N$ の複数のプロセッサ HW_1, HW_2, HW_3, HW_4 に対する割り当て（マッピング）の例を示す。この例の場合、プロセッサの数は4である。しかし、アプリケーションによっては数十個のプロセッサが必要になる。複数のソフトウェア機能が集合して、例えば、レーダシミュレーションや処理プログラムのような独立したプログラムのタスクを構成する。このプログラムは開始タスク1から終了タスク2まで進行する。例えば、プログラムは全体として、場合によっては、合計数万のコードラインによって表わされる数百のタスクを包含する。1つのプロセッサは一時には1つのタスク SW'_k を実行する。図1に示した、ソフトウェアコンポーネントを繋いでいるライン3は、タスクの連続性またはインターフェースを表現したものである。2つのソフトウェアコンポーネントを接続するライン3は、2つのタスクが順次実行されることを示している。つまり、前のタスクが完了したときに次のタスクが実行される。

40

【0009】

プロセッサ HW_1, HW_2, HW_3, HW_4 は、実際の処理回路に加えて、プログラムメモリと、他のハードウェアコンポーネント間のインターフェース回路を有する。プロセッサ HW_k は、例えばカード状である。

【0010】

50

図1は、開始タスク1と終了タスク2の間のデータフローを図示しているが、図示した状態ではソフトウェアコンポーネントの割り当ては最適化されていない。第1の問題は、タスク SW'_k の割り当てがインターフェースと適合していないことである。例えば、タスクの全体的な実行のために分離された2つのタスクが同じプロセッサによって実行されている。当該プロセッサは同時に1つのタスクしか実行できないとすると、開始タスク1から開始タスク2までの全体処理時間は最適化されていない。実際、タスク実行時の1つのプロセッサから他のプロセッサへの入出力は処理を遅延させる。他の問題は、システムが十分な柔軟性を有しないか、あるいはまったく柔軟でないことがある。アルゴリズムの変更の際にはソフトウェアの新規で完全な再配分が必要になるか、あるいは場合によってはハードウェア基本構造の変更さえ必要になる。

10

【0011】

図1は予め設定された規則なしで、4つのプロセッサがプログラムの種々のタスクを実行する流れを示している。プログラムのパートは基本的にプロセッサが利用可能か否かにしたがって割り当てられる。基本タスクが存在するが、プログラムは複数のプロセッサに分配することができるような形で基本タスクに分解されてはいない。タスクによっては2つのプロセッサにまたがるものもある。この割り当て方法は実行するには複雑すぎ、しかも上述のように柔軟性に乏しい。

【0012】

図2は、図1のソフトウェアコンポーネントの割り当てに対応するソフトウェアの基本構造を示すものである。各プロセッサ HW_1, HW_2, \dots, HW_4 に対するソフトウェアレイヤを示す。各プロセッサには実時間オペレーティングシステム(RTOS)が導入されている。このオペレーティングシステムは従来どおりプログラムコード21、22、23の実行を可能にする。このプログラムコードは、詳細規定24に基づくものである。アプリケーションのプログラムコード21、22、23によって定義されたソフトウェアレイヤは、前述のタスク $SW'_1, \dots, SW'_k, \dots, SW'_N$ を全て包含している。

20

【0013】

図3は、本発明の方法を適用して得られたソフトウェア機能の基本構造を示すブロック図である。本発明の方法は、第1の過程でプログラムを基本タスクに分解する。これらのタスクは、ソフトウェアコンポーネントを構成するコードのグループによってプログラム化されている。説明の単純化のために、以降、基本タスクを対応するソフトウェアコンポーネントによって表わすことにする。このような分解は、「コンピュータ支援ソフトウェア操作(Computer-Assisted Software Engineering)」の頭文字をとってCASEと呼ばれるソフトウェア操作システム31によって行う。このCASE装置は、ソフトウェアの構造を決定する、つまり、異なる基本タスク間のリンクあるいは基本タスク相互間の依存関係を定義する。この分解は、基本タスクが可能な限り小さな、つまり、例えば基本タスクを100から200の範囲の可能な限り少ないコードラインを有するソフトウェアコンポーネントに対応させるのが好ましい。CASE装置は、この構造を開始詳細規定21にしたがって作成する。この装置は例えば、基本タスクのリストを定義し、さらに基本タスク相互の接続関係を説明する。このソフトウェア構造に関する情報とタスクのリストはファイル32に格納される。詳細規定は更にファイル33に格納された1つ又は複数のソフトウェア機能を定義する。さらに、利用可能なプロセッサのリストとそれらの相互接続の関係の説明を作成する。このリストは基本タスクによって構成されるプログラムの全体を支持するハードウェア構造を定義する。当該リストはファイル34に格納することができる。これらのファイル32、33、34は、システムの詳細を構成し、後に基本タスクの割り当てのために使用される。

30

【0014】

ソフトウェア構造が定義されると、それぞれのソフトウェアコンポーネントを異なるプロセッサに割り当てる。この割り当て作業は、「従属に関連する割り当てとマッピング(Dependency Related Allocation and Mapping)」の頭文字をとってDRAMと称する第2のソフトウェアツール35によって行う。このソ

40

50

ソフトウェアツールは前述のファイル32、33、34にしたがって、ソフトウェアコンポーネントの最初の割り当てを行う。例えば、この割り当てはランダムに行うことができる。このDRAM装置は、次に、所定の基準に基づいて一連の割り当て評価パラメータのチェックを行う。この基準は例えばデータフロー、プロセッサの負荷、処理時間あるいは設計上の制限に関するものである。チェックしたパラメータが所定の基準を満足しないときは、その割り当てに対してペナルティを加算する。予め設定されたリストに挙げられた全てのパラメータについてチェックを完了すると、DRAM装置はペナルティの合計であるコストを計算する。最適な割り当てとは、コストが最小になる割り当てである。実際には、選択された割り当てのコストは、最小コストではない。いずれの場合にも、割り当てはコストに依存し、コストが高すぎる選択は放棄される。

10

【0015】

ソフトウェアコンポーネントの割り当てとそれに対応するペナルティに関するパラメータチェックの例を以下に説明する。この例の場合には、システムは、それぞれが1つ以上のプロセッサを収容できる4つのカードHW₁、HW₂、HW₃、HW₄を有するものと仮定する。

【0016】

図4に、データフローに関するパラメータチェックのための手段を示す。データフローをチェックするために、第1のカードHW₁への要求の到来によって開始されるパイプライン処理が実行される。詳細には、第1のカードに「トリガー」形式の要求41が送られる。この要求によって第4のカードHW₄の出力においてパラメータ42が起動される。このことはさらに、要求41によって起動された第1のカードHW₁によって処理されたパラメータが、全ての処理を通じて他の全てのカードHW₂、HW₃、HW₄によって処理されることを意味する。不要なデータ転送を最小限に抑えるためには、データは使用直前に処理される必要がある。基本的には要求41によって発生した全ての処理済データが他のカードHW₂、HW₃、HW₄にとって必要であるとの前提の下では、第1のカードHW₁は他のプロセッサとの通信リンク43、44、45を有している。それぞれの基本タスクSW_kに関連して、DRAM装置は、基本タスクSW_kが使用したデータを作成した1つ又は複数の基本タスクを追跡する。データを作成した1つ以上の基本タスクは同一のカードで実行されることも別々のカードで実行されることもありえ、同一のカードであってもプロセッサは同一であることも異なることもありえる。

20

【0017】

データフローを特徴付ける第1のパラメータはプロセッサ相互間の「前方向」伝送である。この場合、対象となる基本タスクの入力データは当該タスクを処理しているカードの物理的にすぐ下流側にあるカードによって処理される。例えば、第3のカードHW₃の基本タスクは第2のカードHW₂が作成したパラメータを必要とする。データ伝送を最小限にするために、入力された全てのパラメータを同一のプロセッサによって処理するか、少なくとも同一のカードによって処理するのが望ましい。カード相互間の前方向データ伝送に対するペナルティは、プロセッサ相互間の伝送に対するペナルティよりも大きい。データが2つ以上のカードを通過する場合にはペナルティは一層大きくなる。したがって、DRAM装置は、例えば、前方向カード間伝送のペナルティにパラメータが作成されてから使用されるまでの間に通過したカードの数を掛けることができる。例えば、前方向のカード間伝送のペナルティは4である。

30

【0018】

データフローに関する第2の基準は後方向カード間伝送である。入力されたパラメータは別のカードで処理され、そのカードは現在のカードよりも物理的に上流側に位置する、つまりデータが未処理状態である。例えば、第2のカードHW₂の基本タスクが、第3のカードHW₃が実行する基本タスクによって作成されるはずのパラメータを必要としている状態である。1つのカードから別のカードへのデータの伝送を最小化するためには、第1のカードHW₁から第4のカードHW₄へ向けて、データは1方向に流れなければならない。後方向のカード間伝送を避けることが重要であれば、後方向伝送に対して大きなペナ

40

50

ルティを与えることができる。このペナルティは例えば 1000 である。

【0019】

考慮しなければならない他の基準はプロセッサの負荷である。DRAM 装置は、同一のプロセッサに与えられるタスクが多すぎないようにプロセッサの負荷を考慮する。例えばタスクのリストを収容したファイルと同じファイルである入力ファイルに、それぞれの基本タスクが必要とするプロセッサへの負荷が定義されている。例えば、ある基本タスクの実行に必要なプロセッサの負荷は、理論的最大利用時、実際的な最大利用時あるいは平均利用時に対して定義される。こうすることによって、プロセッサに対するタスクの適切な割り当てにプロセッサの負荷を直接利用することができる。

【0020】

考慮すべき第 1 の場合は、タスクの実行がプロセッサが許容する、例えば最大負荷の 95 % である閾値を超える場合である。DRAM はこの状態を許容してはならない。95 % の負荷を超えることに対するペナルティはしたがって 10000 と設定される。95 % より低い各状態に対して、ペナルティが次第に小さくなる何段階かの過負荷状態を想定してもよい。プロセッサの負荷が低すぎる場合にペナルティを課すこともできる。これによって利用可能なプロセッサを、当然、過負荷にならない範囲で、最大限に利用することが可能になる。

【0021】

考慮する必要がある他の重要な基準はデータの処理時間である。処理時間は少なくとも 2 つの方法で考慮することができる。チェックする必要がある第 1 の処理時間は、実行頻度の下でプロセッサに割り当てられた全てのタスクを実行するために必要な時間である。実行頻度はタスクごとに定義され、この情報は例えばタスクに関する説明ファイルに格納されている。DRAM 装置は、例えば、1 つのプロセッサがタスクを処理するために必要な処理時間が所定の時間を超えないようにチェックする必要がある。プロセッサの負荷とこの実行時間との間には、必ずしも直接的な関連が存在しなので、このチェックが必要になる場合がある。例えば、1 つのプロセッサの実行時間が所定の時間を超える場合、加算されるペナルティは 10000 である。実行時間が閾値を超える程度に応じて段階的なペナルティを設定することもできる。

【0022】

考慮すべき第 2 の実行時間は全プログラムの実行時間である。各カードに設けられた DRAM 装置によって全てのプログラム実行ブランチが処理される。合計実行時間が最大であるブランチがカードの処理時間を決定する。システム全体によるプログラムの処理時間はそれぞれのカード HW₁、HW₂、HW₃、HW₄ による処理時間の合計である。認められる最大処理時間を超えた場合には、例えば 250000 のような非常に大きなペナルティを加算することができる。認められる処理時間を短縮するか、所定の処理時間の超過に対するペナルティを増加させれば、DRAM 装置はプロセッサを並列駆動させるようになる。

【0023】

チェックすべき他の一連の基準は、システム設計上の制約に関するものである。いろんな理由によってシステム設計者はマッピング、つまりプロセッサに対する基本タスクの割り当てに一定の影響力を行使することを望む場合がある。本発明の方法を実現する方法は一通りではない。特に、設計者は特定のプロセッサに与えられるタスクを規定することができる、この割り当ては例えばシステムの説明ファイルに記述されている。DRAM 装置がこの割り当てには介入しない場合、ペナルティの加算はない。しかし、例えば処理時間やプロセッサの負荷に関する危険な状態が出現した場合には、DRAM 装置は警告メッセージを送信することができる。

【0024】

複数の基本タスクを組み合わせること、つまり割り当てが行われていない同じプロセッサ又は同じカードに複数の基本タスクを割り当てることもできる。複数のタスクが同じデータ群を利用する場合にはこの割り当て方法が特に好適である。このようなタスクの組み合

わせは説明ファイルに記述することができる。このような制限を満足していないことに対するは比較的大きなペナルティ、例えば100を課すことができる。例えば処理時間やプロセッサの負荷のような設計規則に違反することになるために組み合わせが不可能であれば、DRAM装置は警告メッセージを送信する。例えばDRAM装置の定義ファイルには、全てのペナルティが定義されている。

【0025】

ペナルティは慎重に選択するのが好ましい。特に、ペナルティの程度は設計パラメータに与えられた制約の程度に基づいて決定する必要がある。多くの設計規則は相互に関連し、相互に影響し合っている。最適な設計はこれらの最適な組み合わせである。したがって、最適な設計は、ペナルティによって表現されたコストが最小になるものである。本発明によれば、コストを最小化する方法あるいはコストを少なくとも最小値に近づけるための方法は、基本タスクの割り当てアルゴリズムの繰り返しを内容とするものである。割り当てアルゴリズムを繰り返すと、それぞれの解決方法に対して異なるペナルティが課される。繰り返される過程は、

—プロセッサに対して基本タスクの割り当てを行なう過程と、
—予め作成されたリストに掲載されている、割り当てに対する評価パラメータをチェックする過程と、
—割り当てのコストを計算する過程である。

【0026】

好ましい割り当ては複数存在すると考えられる。複数の最適な割り当て相互間では、ペナルティコストは殆ど差が無いので、例えば、閾値として2%から3%の値を選択することができる。繰り返し割り当てを行ってもコストの変化がこの閾値の範囲内であれば、その割り当ては許容できると考えることができる。実際は、コストの変化が収束して、例えば2%から3%の閾値以内になったときにその割り当てを選択する。別のやり方では、繰り返しの間ごとに、1つのタスクをランダムに選択して同様にランダムに選択した別のプロセッサに割り当てる。これらを繰り返してそのたびにコストを計算する。

【0027】

DRAM装置によってタスクの割り当て（マッピング）を行うのに要する総時間は、繰り返し1回当たり5分未満である。したがって、この装置を使って多くの繰り返しを行うことができる。その結果、比較的迅速かつ自動的に、したがって経済的に最適な割り当てを達成することができる。上述のように、ペナルティのコストは、最終的な割り当てが得られたか否かを示す。設計パラメータに対して付与された制約の程度にしたがってペナルティが慎重に選択されていれば、許容することができる解を求めるに当って評価を繰り返すたびにコストの変化は次第に小さくなっていく。コストの変化が大きいことは、1つ以上の割り当てパラメータが根本的に正しくないことを示唆するものである。完全にランダムにマッピングを開始する代わりに、例えばデータフローに関連して定めた所定の基準に基づいてプロセッサを予め選択しても良い。

【0028】

DRAM装置によってタスクの割り当てが完了したら、つまりペナルティコストが許容範囲内になる割り当てが決定されたら、コード発生器36が、使用される複数のプロセッサ又はカードHW₁、HW₂、...、HW₄による全基本タスクの実行を可能にするためのコードを作成する。基本タスクにはソフトウェアコンポーネントSW_kが対応し、数十から数百のコードラインを有する。図3でハッチングを施した部分はコード発生器36が作成したコードに対応する。コード発生器は、プロセッサまたはカードのオペレーティングシステム（RTOS）と相互に通信する中間レイヤ（ミドルウェア）37を作成する。当該コード発生器はさらに各ソフトウェアコンポーネントの周囲にソフトウェアレイヤ38を作成し、ミドルウェアレイヤ37と相互に通信して対応するプロセッサが実行できるようになる。したがって、コード発生器はソフトウェアコンポーネントHW_kとプロセッサを結びつける一種の「接着」コードを作成する。実際、コード発生器は、選択された割り当てにしたがって、ソフトウェアコンポーネントを、DRAM装置が指示する物理的な位

10

20

30

40

50

置、プロセッサおよびカードに結び付ける。

【0029】

図5は、本発明に基づく、ソフトウェアコンポーネント $SW_1, \dots, SW_k, \dots, SW_N$ の、カード HW_1, HW_2, HW_3, HW_4 に対する割り当ての例を示すものである。図では、データフローに関する制約が考慮されている。特に、ソフトウェアコンポーネント SW_k は、カード HW_1, HW_2, HW_3, HW_4 に対して、より適切な割り当てが行われている。特に負荷と実行時間に関する他の制約も当然満足している。図5には本発明の他の利点も示されている。図はメンテナンスが極めて容易になり、同時に信頼性の向上が得られたことを示している。特に、カードが故障した場合、別のカードとのインターフェースの問題がなく、容易に交換可能であることが示されている。システム全体を構成する一部分を置換することも容易かつ経済的に行うことができる。図5に図示した割り当ての例に拠れば、第1の置換部分がハードウェアとソフトウェアを含む第1のカードの機能を分担し、第2の置換部分が第2のカードの機能を分担し、以下同様である。こうして問題なくカードを集合させることができ、カード相互間のハードウェアおよび機能的なインターフェースを確立することが容易になる。

【0030】

最後に、本方法によれば割り当ての柔軟性が大幅に向上する。実際に、1つ以上のタスクを追加する場合、例えばDRAM装置を用いて本発明の方法を適用することは容易である。この場合、開始点は、例えばその時点の構成であり、新たなタスクをランダムに割り当てることができる。繰り返し割り当てを行い、コストが許容できなければ、その結果は新たなプロセッサを追加する必要性を指摘するものであり得る。

【0031】

以上、プロセッサが4つの場合について本発明を説明したが、さらに多くのプロセッサを使用することも可能である。

【図面の簡単な説明】

【図1】複数のプロセッサに対するソフトウェア機能の割り当ての例を示す図である。

【図2】前述の割り当てが行われたソフトウェアの基本構造を示すブロック図である。

【図3】本発明の方法を適用することによって得られたソフトウェアの基本構造を示すブロック図である。

【図4】プロセッサ相互間のデータフローに関するパラメータをチェックする手段の例である。

【図5】本発明の方法を適用して得られたソフトウェア機能の割り当て例を示す図である。

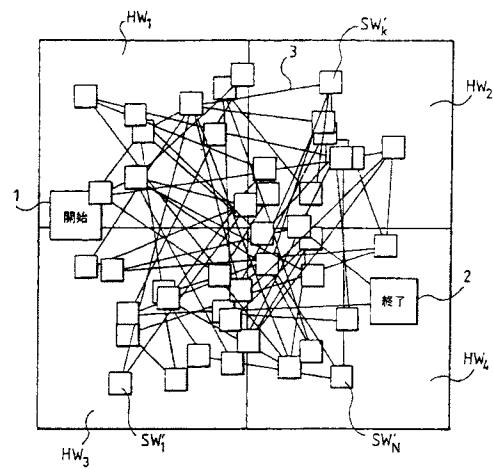
。

10

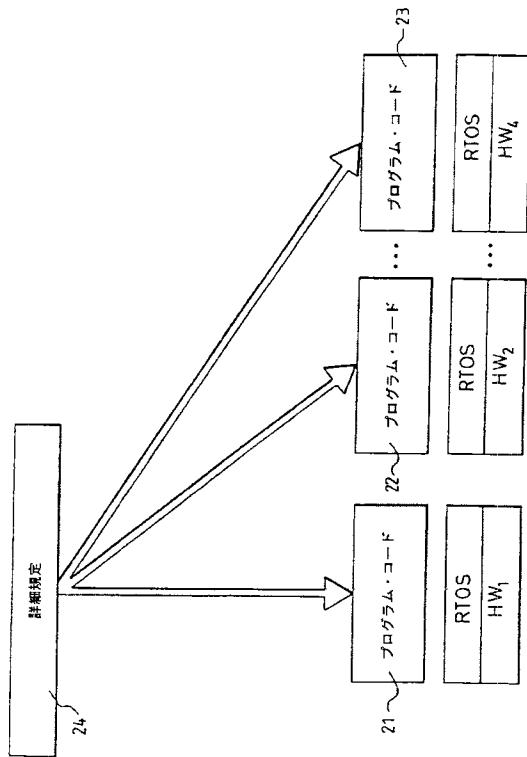
20

30

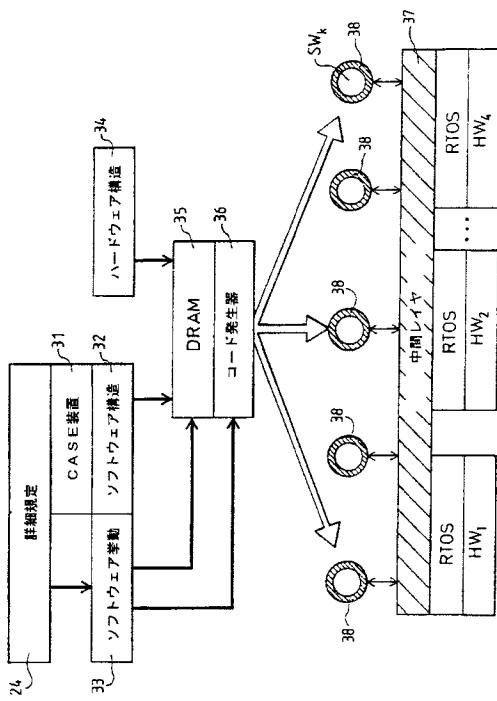
【図 1】



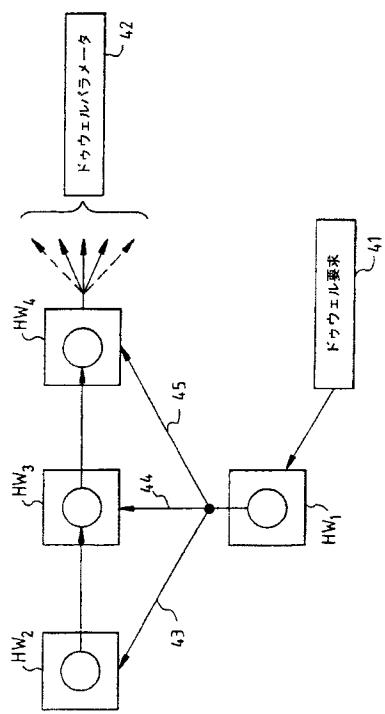
【図 2】



【図 3】



【図 4】



【図5】

